## REMARKS

Claims 5, 14, 31 and 44 are amended. Claims 1-47 remain in the application for consideration. In view of the following remarks, Applicant respectfully requests the application be forwarded to issuance.

### §112 Rejections

Claims 5-7, 14-19, 31-36 and 44-47 stand rejected under 35 U.S.C § 112, second paragraph as being indefinite. Specifically, claims 5-7, 14, 31 and 44 are rejected because the Office argues that the term "receiving an XML request" can be interpreted in two possible ways and therefore is indefinite. The Office argues that this phrase could mean either that the request is in XML or that it is a request for XML.

Claims 5, 14, 31 and 41 have been amended to clarify that the XML request is a request for an XML document. Accordingly, the Office's rejection is traversed.

With respect to claim 14, the Office further argues that the claim is indefinite because the term "in a manner in which" is a relative term and is not defined by the claim. Further, the Office argues that the specification does not provide a standard for ascertaining the requisite degree apparently embodied by this relative term. Applicant respectfully disagrees and traverses the Office's rejection. Applicant respectfully directs the Office to the specification, starting on page 8, line 3 through page 10, line 16, the entirety of which is provided below for the Office's convenience.

> At step 100 a server, such as server 20 (Fig. 2) receives an XML client request. The server 20 prepares only a portion of a response (Step

102) to the client's request. The server 20 then sends the response portion (step 104) to the client. The server 20 then determines whether the response preparation is complete (step 106). If it is not, then step 106 loops back to step 102 and the server 20 prepares another portion of the response. Accordingly, the server 20 repeats steps 102 and 104 until a complete response is sent to the client, at which time the processing for that response ends (step 108).

This constitutes a highly desirable and timely improvement over past methods which required that the entire XML response be built and saved in memory before it was sent to the client. One of the advantages of the present response processing becomes apparent in the context of very large responses that must be prepared for certain client requests. An example of such a response is called a "multistatus" response which is discussed in more detail below. ***By sending a response to a client in a piecewise manner, the client can begin processing the response portions (e.g. parsing the response portions and providing the data to the client application 12) sooner that it could if the server had to build the entire response, save it in memory, and send it out.*** This, in turn, translates to improved processing speeds and reductions in the overhead processing that is necessary to prepare and send the responses.

Fig. 4 shows some exemplary inventive programming structures that can be utilized to generate and send the individual client response portions. Collectively, these programming structures provide an XML response generator. In this example, the response generator includes a request method object 110, an emitter object 112 and a body object 114. ***These objects work together to generate and send response portions to a client in a piecewise manner.*** In the described embodiment, ***there is a request method object 110 for each type of request that can be received from a client. Client request types are defined in terms of the HTTP verbs that are used in the request. When a request is received, the type of request is determined and then an appropriate request method object, e.g. object 110, is created.*** The request method object 110 performs data gathering functions in which it gathers the appropriate data to include in the client's response. Accordingly, request method object 110 constitutes but one example of a data-gathering mechanism. ***When the data is gathered by the request method object 110, a series of calls are made to the emitter object 112. The calls include the data that has been gathered and tell the emitter object 112 the information it needs to do its job.*** The request method object 110 also knows what element tags or nodes that it needs in its response. This information is also conveyed to the emitter object 112 in the calls that are made by the request method object 110. The emitter object

112 is then responsible for formatting the gathered data into an appropriate XML syntax for the response. Accordingly, the emitter object 112 constitutes but one example of a data-formatting mechanism.

In this example, the request method object 110 does not have to know anything about the syntax of the response that is going to be built by the emitter object 112. It only needs to know the information that is necessary for the response, e.g. the XML nodes, their organization and order within the XML response, any text values that are to be included in the response, and the like. Since there is a request method object 110 for each type of client request that can be received, these objects only have to know the information or data that is associated with their particular type of client request. ***In this example, the emitter object 112 is primarily a mechanism by which the information or data is placed into the correct syntactic format.*** Thus, the emitter object 112 does not have to do any data gathering because the data and all other information it needs is provided to it by the request method object 110.

***When the emitter object 112 formats the response portions, it provides the response portions to the body object 114.*** In this example, the body object 114 is a response-sending mechanism that manages the sending function in which the response portions are sent to the client. The body object 114 can also perform other functions such as setting up so-called boiler plate portions of the response (e.g. an XML prologue) that is to be sent. The body object 114 can also accumulate response portions and send them to the client at an appropriate time.

***Thus, the XML response generator is able to generate and send response portion to a client in a piecewise fashion.*** This avoids having to build and save an entire hierarchical tree structure that represents the response document.

The excerpt provided above is replete with an explanation of formatted data which is emitted in a manner in which an XML response can be sent to the client as recited in the claim. The excerpt describes an exemplary architecture, its constituent parts, and how those constituent parts work together to emit formatted data in a manner in which an XML response can be sent to the client. As such, Applicant respectfully submits that a person of skill would indeed be reasonably

apprised of the scope of the claimed subject matter. Accordingly, Applicant respectfully traverses the Office's rejection.

**§103 Rejections**

Claims 1-7, 10-11, 13-14, 16-17, 19, 31-32, 34-35 and 38 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,012,098 to Bayeh et al. (hereinafter "Bayeh") in view of a document entitled "Internet Explorer 5 and XML by Heinemann (hereinafter "Heinemann") in further view of a document entitled "XML Fragment Interchange, W3C Working Draft" (hereinafter "W3C").

Claims 8, 9, 18 and 33 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh, Heinemann, W3C in further view of a document entitled "Extensions for Distributed Authoring..." by Goland et al. (hereinafter "Goland").

Claims 20 and 30 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh in view of Goland.

Claims 21-22 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh in view of Goland in further view of Heinemann.

Claims 23, 25 and 44-47 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh in view of Goland in further view of W3C.

Claims 24, 26 and 41-43 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh in view of Goland in further view of W3C and Heinemann.

Claims 12, 15, 36, 39, and 40 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh, Heinemann and W3C in further view of U.S. Patent No. 6,366,947 to Kavner.

Claims 27-29 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Bayeh in view of Goland in further view of W3C and Kavner.

Claim 37 stands rejected under §103(a) over Bayeh in view of Heinemann.

Before undertaking a discussion of the Office's application of Bayeh and the other references, the following discussion entitled "§103 Standard" is provided. Following a discussion of the §103 standard, a discussion of Bayeh is provided to aid the Office in appreciating some of the differences between the subject matter described in the cited references and the various claimed embodiments.

**The §103 Standard**

To establish a *prima facie* case of obviousness, three basic criteria *must* be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992); *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Second, there must be a reasonable expectation of success. *In re Merck & Co., Inc.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). Finally, the prior art reference (or references when combined) must teach or suggest *all* the claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

Hence, when patentability turns on the question of obviousness, the search for and analysis of the prior art includes evidence relevant to the finding of whether there is a teaching, motivation, or suggestion to select and combine the references relied on as evidence of obviousness. See, e.g., *McGinley v. Franklin*

*Sports, Inc.*, 262 F.3d 1339, 1351-52, 60 USPQ2d 1001, 1008 (Fed. Cir. 2001) ("the central question is whether there is reason to combine [the] references," a question of fact drawing on the Graham factors).

"The factual inquiry whether to combine references must be thorough and searching." *Id.* It must be based on objective evidence of record. This precedent has been reinforced in myriad decisions, and cannot be dispensed with. See, e.g., *Brown & Williamson Tobacco Corp. v. Philip Morris Inc.*, 229 F.3d 1120, 1124-25, 56 USPQ2d 1456, 1459 (Fed. Cir. 2000) ("a showing of a suggestion, teaching, or motivation to combine the prior art references is an 'essential component of an obviousness holding'") (quoting *C.R. Bard, Inc., v. M3 Systems, Inc.*, 157 F.3d 1340, 1352, 48 USPQ2d 1225, 1232 (Fed. Cir. 1998)); *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999) ("Our case law makes clear that the best defense against the subtle but powerful attraction of a hindsight-based obviousness analysis is rigorous application of the requirement for a showing of the teaching or motivation to combine prior art references."); *In re Dance*, 160 F.3d 1339, 1343, 48 USPQ2d 1635, 1637 (Fed. Cir. 1998) (there must be some motivation, suggestion, or teaching of the desirability of making the specific combination that was made by the applicant); *In re Fine*, 837 F.2d 1071, 1075, 5 USPQ2d 1596, 1600 (Fed. Cir. 1988) ("'teachings of references can be combined only if there is some suggestion or incentive to do so.'") (emphasis in original) (quoting *ACS Hosp. Sys., Inc. v. Montefiore Hosp.*, 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984)).

The need for specificity pervades this authority. See, e.g., *In re Kotzab*, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) ("particular findings must be made as to the reason the skilled artisan, with no knowledge of the

claimed invention, would have selected these components for combination in the manner claimed").

In view of the above-described Standard, Applicant respectfully submits that the Office has failed to establish a *prima facie* case of obviousness.

**The Bayeh Reference**

Bayeh is directed to a method and system that utilizes servlet pairing for isolation of the retrieval and rendering of data. In accordance with its disclosure, data retrieval logic is isolated to a data servlet, and presentation formatting is isolated to a rendering servlet. Servlet chaining is used to send the output of the data servlet to the rendering servlet. The data servlet formats its output data stream for transfer to a downstream servlet. This data stream is formatted using a language such as the Extensible Markup Language (XML), according to a specific Document Type Definition (DTD). The rendering servlet parses this XML data stream, using a style sheet that may be written using the Extensible Style Language (XSL), and creates a HyperText Markup Language (HTML) data stream as output.

Bayeh describes the processing that takes place on the server side as follows:

> At Step 230, the server which received the client's request routes it to the proper data servlet. ***
>
> Steps 240 through 270 are implemented by a data servlet according to the present invention. These steps represent the logic associated with data retrieval, as well as minimal formatting of the data for transfer to a rendering servlet: the formatting is not a presentation format.

At Step 240, the data servlet processes the client request. The request will typically require retrieving data from some database available to the data servlet. The data servlet will format a database query request, using an appropriate query language that will depend on the type of database on which the relevant data is stored. ***

Step 250 is included to indicate that, optionally, additional processing may be performed on the data received in response to the database query. ***

At Step 260, the data servlet formats the database information as an XML data stream. As previously discussed, a DTD is used in this formatting step. The DTD specifies how specific predefined "tags" are to be inserted into the XML data stream. A tag is a keyword that identifies what the data is which is associated with the tag, and is typically composed of a character string enclosed in special characters. "Special characters" means characters other than letters and numbers, which are defined and reserved for use with tags. Special characters are used so that a parser processing the data stream will recognize that this a tag. A tag is normally inserted preceding its associated data: a corresponding tag may also be inserted following the data, to clearly identify where that data ends. ***

When the entire XML data stream required for the database results has been formatted by the data servlet, that data stream is sent on to the next servlet in the chain at Step 270. In the preferred embodiment, the next servlet is the rendering servlet.

Steps 280 through 320 are implemented by a rendering servlet according to the present invention. These steps represent the logic associated with data presentation formatting.

Step 280 receives the XML data stream created by, and sent by, the data servlet. Techniques for receiving a data stream from a chained servlet are well known in the art, and Step 280 is shown for completeness of depicting the function of the rendering servlet.

According to the preferred embodiment, the rendering servlet ***must*** parse the XML data stream, and reformat it into HTML. ***This is necessary*** because browsers, by convention, expect to receive data that has been formatted with HTML. As discussed previously, this parsing process requires two types of data input: the XML data stream, and style sheet information. In the preferred embodiment, an XSL style sheet is used.

Techniques for writing parsers are well known in the art, and will not be described in detail herein.

Step 290 indicates that the rendering servlet locates the XSL style sheet. Typically, the style sheet will be stored as a file on a medium, such as a disk drive, accessible to the rendering servlet. Alternatively, the style sheet might be incorporated directly into the code of the rendering servlet; however, because incorporating the information directly into the code makes revising the style sheet more difficult, this alternative is less desirable than isolating the style sheet as a separate file.

At Step 300, the rendering servlet parses the XML data stream, using the input sources described above. A parser reads a data stream, looking for predefined strings of characters that the parser recognizes, and which indicate to the parser what type of data is represented. In the preferred embodiment, the DTD used in the data servlet specified predefined strings that were used as tags, as described above, and inserted into the XML data stream by the data servlet. The parser looks for these tags as it reads the XML data stream. When a recognized tag is found, the parser knows what type of XML document element appears in the data stream between this tag and its corresponding ending tag (or following this tag, if ending tags are not required). As the parser in the rendering servlet determines what each document element is, it creates a new data stream, formatted using HTML. The parser may also insert presentation style attributes into the HTML data stream, where the appropriate attribute to use is determined according to the type of document element the parser is currently processing. Creation of the HTML data stream is represented in FIG. 5 as Step 310, although one skilled in the art will recognize that the functions of Steps 300 and 310 are intermingled. That is, as the parser processes portions of the input XML data stream, it creates the corresponding HTML data stream, then processes another portion of the input data stream, creates more of the output data stream, etc., until the entire input data stream has been processed.

***When the reformatted data stream is complete, Step 320 sends that HTML data stream back to the client's browser***, to be processed by the browser for presentation to the user.

**Applicant's Disclosure**

Applicant's disclosure describes various methods and systems for generating and sending XML documents and, in particular, generating and sending an XML response to an XML client request.

In various described embodiments and not necessarily all of the described and claimed embodiments, an XML document is prepared and sent to a client only a portion at a time. XML document portions are generated and sent until an entire XML document is sent to the client. In a specific implementation an instance of which is claimed in one claim set, an XML response generator is provided and responds to a client request without having to first build and save a hierarchical tree structure in memory that represents the response. The response generator includes one or more request method objects. There is, in this embodiment, one request method object for each particular type of client request that might be received. Each request method object knows and gathers the data that is needed to respond to its particular associated client request. In addition, the request method object knows the order in which the information must be provided.

In accordance with one embodiment, the request method object calls an emitter object with the data that is gathered by the request method object. The calls are made in a particular order and ensure that the hierarchical nature of the response that is being built is preserved. The emitter object translates the data that it receives into response portions that are in proper XML syntactic form.

In accordance with one embodiment, a body object is provided to manage a buffer. The emitter object calls the body object with the properly-formatted XML response portions. The response portions are placed in the buffer. When a defined buffer threshold is reached, the buffered response portions are sent to the client.

**The Claims Rejections**

**Claim 1** recites a method of generating an XML document comprising:

- preparing ***only a portion*** of an XML document;
- sending said portion to a client; and
- ***repeating said preparing and said sending*** until an entire XML document is sent to a client.

In making out the rejection of this claim, the Office notes that Bayeh discloses processing and formatting results as XML, and sending a document as HTML. The Office admits that Bayeh does not teach sending XML, but rather HTML. The Office then relies on Heinemann which discusses Internet Explorer 5 and its support of XML Schema, XSL and the like. Based on this, the Office argues that it would be obvious to combine the teachings of Bayeh and Heinemann to send XML to a client as it would allow the client computer to use the capabilities of XML such as validation. Applicant does not understand the Office's reasoning as it completely disregards Bayeh's teaching of sending HTML to a client. Applicant respectfully disagrees with the Office because this directly contradicts Bayeh's teachings.

Bayeh specifically teaches a division of labor between data retrieval and data presentation. See, e.g. column 3, lines 62 through column 4, lines 22. Bayeh's systems and methods retrieve data using a first servlet 83, use XML to format the structured data, provide the XML data to a second servlet 85 which converts the XML data into a presentation format comprising HTML, which is then sent on to the client. Bayeh specifically teaches, in connection with its

division of labor between data retrieval and data presentation, that the XML format is used as a formatting notation for the retrieved structure data (see e.g., column 8, lines 13-25), and that this retrieved structured data is then converted into a presentation format comprising the HTML (see, e.g., column 8, lines 29-34).

Bayeh further instructs on the advantages of isolating the function of the rendering servlet 85 from the function of the data servlet 83, starting in column 8 at line 36. For the convenience of the Office, this excerpt is reproduced in its entirety below:

> By isolating the function of the rendering servlet 85 from the function of the data servlet 83, the advantages of structured, modular programming are achieved. As discussed earlier, ***one of these advantages is simplifying the change process*** if changes are required to either the data retrieval logic (isolated in the data servlet 83), or to the data presentation formatting logic (isolated in the rendering servlet 85). ***Further, system throughput can be optimized*** by having multiple data servlets 83 and multiple rendering servlets 85: if one rendering servlet 85 is busy, the data servlet 83 does not have to wait to use the rendering services--it can simply pass the data to be rendered to a different rendering servlet 85'. ***This prevents bottlenecks in the system***, where one process is delayed by another process. ***System performance is also optimized in this model***, because decoupling the data retrieval logic from the presentation formatting logic allows each servlet to be optimized in its functionality. For example, a unique data servlet 83' might be created to retrieve data from a specific type of database 88 used by the server 82. Similarly, unique rendering servlets 85 might be created to format data according to different presentation requirements. When a servlet is written to retrieve data from a specific, known database (equivalently, to format data for a specific, known environment), the servlet code can be optimized for that use. ***Additionally, system flexibility is optimized*** because the unique servlets can function as pluggable components, which come and go as the needs of the environment change.

In the excerpt above, there are at least four advantages that Bayeh instructs are achieved by its system. These advantages are not insignificant to Bayeh, as Bayeh uses terms such as "optimized" to describe the results that it achieves. Yet, if Bayeh is modified in, as the Office contends, an obvious manner, to send "unformatted XML", doing so would completely eliminate the rendering servlet 85 (Fig. 4) which Bayeh describes as a necessary component to achieve all of its optimizations.

To this extent, Bayeh teaches directly away from sending XML to a client. Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness.

In addition, the Office admits that neither Bayeh nor Heinemann disclose "dealing with the XML in portions." The Office then argues that W3C discloses a method of dividing XML into fragments and sending them, citing to the Abstract for support. Based on this, the Office argues that it would be obvious to combine the teachings of W3C with the above combination "to deal with portions of XML so that if the user wants a particular section, he does need to receive them all."

Applicant respectfully disagrees with respect to the Office's attempted combination and respectfully traverses the Office's rejection. W3C states, in the Abstract section, that "[t]he XML Fragment WG is chartered with defining a way to send fragments of an XML document—regardless of whether the fragments are predetermined entities or not—***without having to send all of the containing document up to the part in question***." Applicant respectfully directs the Office's attention to the last element of claim 1 which recites, "...repeating said preparing and said sending ***until an entire XML document is sent to a client***." Thus, to this extent, W3C teaches directly away from the subject matter recited in this claim.

Accordingly, for the reasons mentioned above, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 2-4** depend either directly or indirectly from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither shown nor suggested in the references of record, either singly or in combination with one another.

**Claim 5** recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving a request from a client for an XML document;
- preparing only a portion of a response to the request; and
- sending the response portion to the client.

In making out this rejection, the Office argues that this claim is rejected based on the combination of Bayeh, Heinemann and W3C. As noted above, the Office has failed to establish a *prima facie* case of obviousness with respect to the combination of Bayeh and Heinemann. Specifically, it is irrelevant that Heinemann discusses Internet Explorer 5 insofar as it supporting XML, when one considers the specific context of Bayeh and what Bayeh teaches. There is no support whatsoever in Bayeh for sending anything to the client other than HTML. Modifying Bayeh, as argued by the Office, is contrary to Bayeh's teachings. Hence, the Office has failed to establish a *prima facie* case of obviousness.

In addition, the Office admits that neither Bayeh nor Heinemann disclose dealing with XML in portions. The Office then relies on W3C and argues that is discloses a method of dividing XML into fragments and sending the fragments.

Based on this teaching, the Office argues it would be obvious to combine the teachings of these three references to render the subject matter of claim 5 obvious. Applicant respectfully disagrees and submits that the Office has failed to establish a *prima facie* case of obviousness. Specifically, consider the language of claim 5 which is provided below:

- receiving a request from a client ***for an XML document***;
- preparing only a portion of a response to the request; and
- sending the response portion to the client.

Now consider W3C, particularly the subject matter appearing on page 3, 2nd paragraph which was cited by the Office in support of the present rejection. Specifically, W3C states:

> In the case of many XML documents, it is suboptimal to have to receive and parse the entire document when only a fragment of it is desired. ***If the user asked to look at chapter 20***, one shouldn't need to parse 19 whole chapters before getting to the part of interest.

Thus, being consistent with the claim language, if one considers what the W3C user asks for as "an XML document", then it is clear that W3C does not teach sending only a portion of chapter 20 to the client. Rather, W3C teaches sending the entire chapter 20 to the user. It just so happens that in this case, the XML document requested by the user (i.e. chapter 20) is part of a larger document that the user has not apparently asked for. To this extent, W3C teaches directly away from the subject matter of claim 5. For this additional reason, the Office has failed to establish a *prima facie* case of obviousness.

**Claims 6-13** depend either directly or indirectly from claim 5 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 5, are neither shown nor suggested in the references of record, either singly or in combination with one another. Additionally, **claims 8 and 9** are rejected over a further combination with Goland, and **claim 12** is rejected over a further combination with Kavner. Given the Office's failure to establish a *prima facie* case of obviousness with respect to the combination of Bayeh, Heinemann and W3C, the Office's reliance on Goland and Kavner is misplaced and is not seen to add anything of significance.

**Claim 14** recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving a request from a client for an XML document;
- gathering data that is to appear in a response to the client's request;
- calling an emitter object and passing the emitter object the gathered data;
- formatting the gathered data into an appropriate XML syntax with the emitter object; and
- emitting formatted data from the emitter object, the emitter object emitting the formatted data in a manner in which an XML response can be sent to the client without having to build a hierarchical tree that represents the XML response.

In making out the rejection, the Office argues that Bayeh discloses receiving a request and cites to column 10, lines 19-25 in support therefore. While Bayeh does disclose receiving a client request, it does not disclose receiving a request from the client for an XML document. The Office also argues that Bayeh discloses gathering data, formatting the data into XML, and emitting formatted

data. Applicant notes that Bayeh does not disclose or suggest emitting an XML response to a client. Rather, Bayeh *explicitly* teaches and leaves no room for sending anything other than an HTML response to the client.

The Office then argues that Bayeh does not teach formatting with the object that was passed the data. Rather, the Office notes, Bayeh teaches that first it is formatted within the same object that gathered the data. The Office then argues that it would have been obvious to have one object do both functions as it would reduce communications between objects. Modifying Bayeh as proposed by the Office contradicts the express teachings of Bayeh which specifically teach that the data servlet that gathers the data also formats the data for the next data servlet. Accordingly, there is no support for the Office's modification.

The Office further admits that Bayeh does not teach sending XML, but rather HTML. The Office then relies on Heinemann's discussion of Internet Explorer 5 and its support of XML. The Office argues, based on this, that it would be obvious to combine the teachings of Bayeh and Heinemann to send unformatted XML to a client computer. Applicant disagrees. Bayeh teaches directly away from sending XML to a client computer by specifically teaching that HTML is sent to the client after conversion from XML. Accordingly, the Office has not established a *prima facie* case of obviousness with respect to the combination of Bayeh and Heinemann.

Further, the Office notes that Bayeh makes no mention of building a hierarchical tree. As such, the Office interprets its absence to mean that it also emits data in a manner in which a tree would not have to be built. Applicant disagrees strongly with the Office's logic, particularly in view of the discussion in Bayeh appearing in column 8, lines 3-29. There, Bayeh discusses the role of data

servlet 83 insofar as formatting the data in XML so that a downstream servlet (which expects its input in XML format—see, e.g. lines 23-26) can receive and process the XML. Thus, by virtue of the fact that servlet 83 emits an XML stream in the XML format, it likely does build a hierarchical tree as XML is inherently a hierarchically tag-based language.

Notwithstanding this misinterpretation of Bayeh, the Office further argues that W3C discloses sending XML fragments and, as such, none of the portions would be the full XML response. Based on this, the Office reasons that there would be no need to build the hierarchical tree that represents the full XML response. As such, the Office concludes that it would be obvious to combine Bayeh, Heinemann, and W3C to render the subject matter of this claim obvious. Applicant respectfully disagrees. Applicant can find no disclosure whatsoever in W3C that even remotely suggests that hierarchical trees are not used in building a client response. If the Office is going to take the position that W3C's fragments eliminate the need to build a hierarchical tree is formulating a client response, Applicant respectfully requests the Office to point to a specific portion of the W3C that spells this out. Short of finding a specific discussion in W3C along these lines, it is inappropriate to assume that a reference does not have properties just because those properties are not described.

Accordingly, for at least this additional reason, this claim is allowable.

**Claims 15-19** depend directly from claim 14 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 14, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the

references to Kavner and Goland are not seen to add anything of significance to the rejections of **claim 15 and 18** respectively.

**Claim 20** recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving an XML request from a client, the XML request containing a Web Distributed Authoring and Versioning (WebDAV) request method;
- ***determining the WebDAV request method*** that is contained in the client's request;
- ***creating a request method object for the WebDAV request method***;
- gathering data that is to appear in a response to the client's request with the request method object;
- calling an emitter object and passing the emitter object data that was gathered by the request method object; and
- generating at least a portion of a syntactically correct XML response with the emitter object using the data that was gathered by the request method object.

In making out the rejection of this claim, the Office argues that Bayeh discloses receiving an XML request, and gathering data for a response with an object. The Office then notes that Bayeh does not disclose calling and passing data to another object that would generate the XML. The Office further admits that Bayeh does not disclose the request being a WebDAV method. The Office then relies on Goland and argues that it discloses several WebDAV request methods. The Office then apparently concludes "[a]s the methods have different functions, it would be inherent to determine what method is contained before processing. It would have been obvious to one of ordinary skill in the art at the time of the invention to request with WebDAV as any data gathering method could be used." Applicant respectfully submits that this has nothing to do with the

subject matter recited in this claim. Specifically, the claim recites, in pertinent part:

> ***determining the WebDAV request method*** that is contained in the client's request;
> - ***creating a request method object for the WebDAV request method***;

None of the references singly or in combination disclose or teach this feature. Accordingly, for at least this reason, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

Accordingly, for all of these reasons, this claim is allowable.

**Claims 21-30** depend directly or indirectly from claim 20 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 20, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the references to Heinemann, W3C and Kavner are not seen to add anything of significance to the rejection of those claims that are further rejected over them.

**Claim 31** recites an Extensible Markup Language (XML) request processor comprising:

- an XML response generator comprising:
  - a request-receiving mechanism configured to receive a request from a client for an XML document;
  - a response-preparing mechanism coupled with the request-receiving mechanism and configured to prepare ***only a portion of a response at a time***; and
  - a sending mechanism coupled with the response-preparing mechanism and configured ***to receive response portions from the response-preparing mechanism and to send the response portions to the client, the sent response portions constituting less than an entirety of a response***.

In making out the rejection of this claim, the Office argues that Bayeh discloses receiving a request and preparing a response. The Office admits that Bayeh does not teach sending XML to a client, but relies on Heinemann for this feature. The Office then admits that neither Bayeh nor Heinemann disclose dealing with XML in portions. The Office then relies on W3C and argues that it discloses sending a method of dividing XML into fragments which constitute less than entirety of the documents and well as sending the fragments. Based on these teachings, the Office argues that it would be obvious to combine the teachings of these references to render the subject matter of this claim obvious.

Applicant disagrees and submits that the Office has not established a *prima facie* case of obviousness for a couple of different reasons. First, there is no motivation or foundation in Bayeh to support modifying it to provide XML responses to a client. In point of fact, Bayeh teaches directly away from this notion. Second, the claim at issue recites that the sent response portions constitute *less than an entirety of a* ***response***. W3C, on the other hand, teaches sending less than an entirety of a ***document*** to a client, with the portion that is sent constituting the complete response. These are two different things. To this extent, W3C teaches directly away from the presently-claimed subject matter. As such, the Office has failed to establish a *prima facie* case of obviousness and this claim is allowable.

**Claims 32-36** depend directly from claim 31 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 31, are neither shown nor suggested in the references of record, either singly or in

combination with one another. In view of the allowability of claim 31, the references to Goland and Kavner are not seen to add anything of significance to the rejection of those claims that are rejected over them.

**Claim 37** recites an Extensible Markup Language (XML) request processor comprising:

- a data-gathering object for gathering data that is to appear in a client response and generating calls in a predefined order that contain the gathered data; and
- an emitter object configured to receive calls that are generated by the data-gathering object and format the data contained therein into an appropriate XML syntax.

In making out the rejection of this claim, the Office argues that Bayeh discloses gathering data and formatting the data into XML with a data servlet. The Office further argues that Bayeh discloses emitting formatted data. The Office admits that Bayeh does not teach formatting with the object that was passed the data, but rather teaches that the data is first formatted within the same object that gathered it. The Office further argues that "in the case of only one call, the pre-defined order of the calls would be one."

The Office further admits that Bayeh does not teach sending XML and then relies on Heinemann for this feature. Based on this, the Office argues that it would be obvious to combine the teachings of these two references to render the subject matter of this claim obvious. Applicant respectfully disagrees and traverses the Office's rejection.

Applicant respectfully points out that the claim at issue recites, *inter alia,* "generating calls in a ***predefined*** order". That is, the claim element recites "calls"

in the plural – meaning that more than one call is generated and that the calls are generated in a predefined order.

As neither Bayeh nor Heinemann disclose anything of the sort, the Office has failed to establish a *prima facie* case of obviousness.

**Claims 38-40** depend either directly or indirectly from claim 37 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 37, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the references to W3C and Kavner are not seen to add anything of significance to those claims rejected in view of these references.

**Claim 41** recites a computer-readable medium having a computer program for responding to an XML request, the program comprising the following steps:

- receiving a client request;
- ***determining an HTTP verb*** that is contained in the client request;
- instantiating a request method object that ***corresponds to*** the HTTP verb that is contained in the client request;
- using the request method object to gather information that is to appear in a response to the client's request;
- making a series of calls to an emitter object that is configured to receive information from the request method object and process the information into a ***response portion*** having an appropriate XML syntactic format; and
- ***sending the response portion*** to the client.

The Office argues that Bayeh discloses receiving a request, gathering data, and emitting formatted data. The Office admits that Bayeh does not teach formatting with the object that was passed the data. The Office further admits that Bayeh does not teach HTTP verbs. The Office then relies on Goland's disclosure

of WebDAV methods and interprets these to be HTTP verbs. The Office goes on to admit that neither Bayeh nor Goland teach calling an object multiple times or dealing with portions of XML. The Office then relies on W3C and argues that it discloses methods of dividing XML into fragments and sending the XML fragments to a client. The Office then notes that Bayeh, Goland and W3C do not teach sending XML and relies on Heinemann for this feature.

Based on all of these teachings, the Office argues that it would be obvious to combine these teachings to render the subject matter of this claim obvious. Applicant respectfully disagrees and traverses the Office's rejection.

Applicant submits that the Office has not even considered all of the subject matter recited in this claim. For example, the claim recites, *inter alia,* "determining an HTTP verb that is contained in the client request; ***instantiating a request method object that corresponds to*** the HTTP verb that is contained in the client request; using the request method object to gather information that is to appear in a response to the client's request...." The cited references fail to disclose or even suggest these features. Accordingly, for at least this reason, this claim is allowable.

**Claims 42 and 43** depend directly from claim 41 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 41, are neither shown nor suggested in the references of record, either singly or in combination with one another.

**Claim 44** recites a computer-readable medium having software code that is configured to receive a request from a client for an XML document and ***instantiate an object that corresponds to an HTTP verb*** that is contained in the

request. The software code further uses the object to build a portion of an XML response to the request.

This claim is rejected by the Office over the combination of Bayeh, Goland, and W3C. Applicant submits that none of the references singly or in combination with one another disclose or teach software code that is configured to receive a request from a client for an XML document and instantiate an object that ***corresponds to an HTTP verb that is contained in the request***. Accordingly, the Office has failed to make out a case of *prima facie* obviousness and this claim is allowable.

**Claims 45-47** depend directly from claim 44 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 44, are neither shown nor suggested in the references of record, either singly or in combination with one another.

## Conclusion

All of the claims are in condition for allowance and Applicant respectfully requests a Notice of Allowability be issued forthwith. In the event that the Office's next action is anything other than issuance of a Notice of Allowability, Applicant respectfully requests that the undersigned be contacted for the purpose of scheduling an interview.

Respectfully Submitted,

Dated: 11/13/03

By: _____

Lance R. Sadler
Reg. No. 38,605
(509) 324-9256